PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-207200

(43)Date of publication of application : 28.07.2000

(51)Int.Cl. G06F 9/06
G06F 12/14

(21)Application number : 11-361095

(22)Date of filing : 20.12.1999

(71)Applicant : NETACTIVE INC

(72)Inventor : GORDON EDWARD RAROOZU
SCOTT ALAN TOMSON

(30)Priority

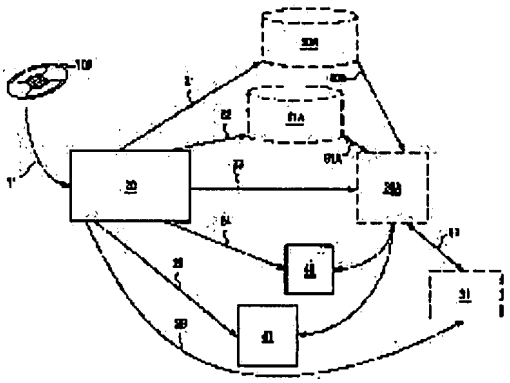
Priority number : 98 218413 Priority date : 22.12.1998 Priority country : US

(54) SOFTWARE PROGRAM PROTECT SYSTEM AND MACHINE-READABLE MEDIUM

(57)Abstract:

PROBLEM TO BE SOLVED: To prevent revision and redistribution of unapproved computer software by reconverting the converted data inversely by using a conversion key and sending the data in an unconverted state to an execution program.

SOLUTION: Converting file sets 50A and 51A are accessed by a corrected execution program 30A via an interface reading/writing function 61A and an interface read-only function 60A. These functions provide proper run-time conversion and inverse conversion. When there is file operation hindrance, a file converting module program retrieves converted data from at least one converted data file, coverts the converted data inversely into its unconverted state by using a conversion key, and sends the data in the unconverted state to an executing program.



LEGAL STATUS

[Date of request for examination] 23.01.2001

b)

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2000－207200
（P2000－207200A）
(43)公開日　平成12年７月28日(2000.7.28)

(51)Int.Cl.'	識別記号	FI		テーマコード'(参考)
G06F 9/06	550	G06F 9/06	550Z	
12/14	320	12/14	320F	

審査請求　未請求　請求項の数46　OL　（全 16 頁）

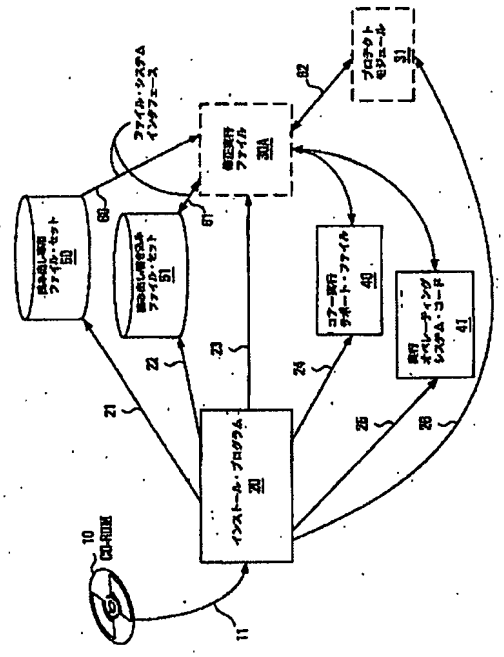
(21)出願番号　特願平11－361095
(22)出願日　平成11年12月20日(1999.12.20)
(31)優先権主張番号　09／218413
(32)優先日　平成10年12月22日(1998.12.22)
(33)優先権主張国　米国（US）

(71)出願人　399126743
ネットアクティブ・アイエヌシー
NETACTIVE INC.
カナダ国，ケイ２イー　８シー４　オンタリオ，ネピーン，スイート 530，アンテアーズ ドライブ 1
(72)発明者　ゴードン・エドワード・ラローズ
カナダ国，ケイ２シー　０イー３，オンタリオ，オタワ，ベースライン ロード 2417
(74)代理人　100097216
弁理士　泉　和人　（外１名）

最終頁に続く

(54)【発明の名称】　ソフトウェア・プログラム・プロテクト・システムおよび機械読み出しが可能な媒体

(57)【要約】
【課題】　本発明は、非認可のコンピュータ・ソフトウェア・プログラムの修正・再配布を防止するプロテクト機構を提供する。
【解決手段】　本発明は、ソフトウェア・プログラムのアセット・ファイルとコアー実行プログラム自身の両方を変換するソフトウェア・プロテクト機構を開示する。そこでは、修正された実行プログラムと対応の変換アセット・ファイルを再配布することが必要である。ファイル変換モジュールは、オペレーティング・システムからの全てのファイル・アクティビティを妨害するために使用され、またオペレーティング・システムへの要求が修正実行プログラムへの（からの）変換データに対して起動されたと識別されたときに限り逆変換を実行する。アセット・リスト、呼プロセス識別情報、変換キー、および変換アルゴリズム等を含むプロテクト・モジュールは、ファイル変換モジュールと関連して使用される。



1

【特許請求の範囲】

【請求項１】　（ａ）少なくとも１つの変換データ・ファイルをストアするメモリと、（ｂ）オペレーティング・システム、実行プログラム、および前記の実行プログラムに変換サービスを供給するファイル変換モジュール・プログラムを実行するプロセッサを有し、

前記ファイル変換モジュール・プログラムは、前記実行プログラムから前記オペレーティング・システムへのファイル・オペレーションを妨害し、

ここで、ファイル・オペレーションの妨害時に、前記ファイル変換モジュール・プログラムは、少なくとも１つの前記変換データ・ファイルから変換データを検索し、変換キーを用いて、前記変換データを、その未変換状態に逆変換し、前記データを、未変換状態のまま、前記実行プログラムに送ることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項２】　請求項１記載のシステムにおいて、前記ファイル・オペレーションは、ファイル・オープン・ファイル・オペレーションと、ファイル読み出しファイル・オペレーションと、ファイル書き込みファイル・オペレーションと、ファイル・クローズ・ファイル・オペレーションとを含むことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項３】　請求項２記載のシステムにおいて、ファイル・オープン・ファイル・オペレーションが行われる際、前記ファイル変換モジュール・プログラムは、少なくとも１つの前記変換データ・ファイルに対して、独自のファイル識別子を割り当て、前記独自のファイル識別子を、アクティブ・ファイル・リスト中に記録することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４】　請求項３記載のシステムにおいて、ファイル読み出しおよびファイル書き込みファイル・オペレーションを行う際、前記ファイル変換モジュール・プログラムは、アクティブ・ファイル・リストをチェックして、逆変換を行うべきかどうかを決定し、行わない場合には、前記ファイル・オペレーションを、通常処理用の前記オペレーティング・システムに戻すことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項５】　請求項３記載のシステムにおいて、前記変換データ・ファイルを閉じる際、前記ファイル変換モジュール・プログラムは、前記アクティブ・ファイル・リストから、少なくとも１つの変換データ・ファイルに対する前記独自のファイル識別子を検出することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項６】　請求項１記載のシステムにおいて、少なくとも１つの前記変換データ・ファイルは、読み出し専用ファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

2

【請求項７】　請求項１記載のシステムにおいて、少なくとも１つの前記変換データ・ファイルは、読み出し/書き込みファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項８】　請求項１記載のシステムにおいて、前記実行プログラムは、前記実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼と共に組み込まれ、前記プロテクト・モジュール・プログラムは、前記実行プログラムを最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項９】　請求項８記載のシステムにおいて、前記変換キーは、前記プロテクト・モジュール・プログラム内にストアされることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項１０】　請求項８記載のシステムにおいて、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリ（ＤＬＬ）であることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項１１】　請求項１記載のシステムにおいて、少なくとも１つの前記変換データ・ファイルは、ＤＥＳおよびＲＳＡ暗号化アルゴリズムのうちの１つを用いて暗号化されることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項１２】　請求項１記載のシステムにおいて、非実行ライセンス・データを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項１３】　請求項１記載のシステムにおいて、実行プログラム・コードを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項１４】　実行プログラムと１以上のデータ・ファイルを含むコンピュータ・ソフトウェア・アプリケーションから、コンピュータ・ソフトウェア・アプリケーションのプロテクト版を生成する方法において：

（ｉ）　１以上の変換アルゴリズムと変換キーを用いて、１以上のデータ・ファイルを、１以上の各変換データ・ファイルに変換し、

（ｉｉ）　前記変換キーをストアし、

（ｉｉｉ）　変換サービスを前記実行プログラムに供給するファイル変換モジュール・プログラムを検索し、

（ｉｖ）　前記の１以上の各変換データ・ファイルと、変換キーと、実行プログラムと、ファイル変換モジュール・プログラムとを、ユーザに配布するための媒体に転送するステップからなることを特徴とするコンピュータ

・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項１５】　請求項１４記載の方法において、前記実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼を前記実行プログラムに組み込むことによって前記実行プログラムを修正するステップをさらに含み、

前記プロテクト・モジュール・プログラムは、前記実行プログラムを最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項１６】　請求項１５記載の方法において、前記変換キーをストアするステップは、前記プロテクト・モジュール・プログラム内に前記変換キーを組み込むステップを含むことを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項１７】　請求項１５記載の方法において、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリー（ＤＬＬ）であることを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項１８】　請求項１４記載の方法において、前記変換アルゴリズムは、ＤＥＳおよびＲＳＡ暗号化アルゴリズムのうちの１つから選択されることを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項１９】　請求項１４記載の方法において、非実行ライセンス・データを前記実行プログラム中に組み込むステップをさらに含み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項２０】　請求項１４記載の方法において、実行プログラム・コードを前記実行プログラム中に組み込むステップをさらに含み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とするコンピュータ・ソフトウェア・アプリケーションのプロテクト版生成方法。

【請求項２１】　少なくとも１つの変換データ・ファイルと、少なくとも１つの変換キーと、変換サービスを前記実行プログラムに供給するファイル変換モジュール・プログラムとを含む機械読み出しが可能な媒体において、

前記実行プログラムと前記ファイル変換モジュール・プログラムとがコンピュータ・システ上で走るとき、前記ファイル変換モジュール・プログラムは、前記実行プログラムから前記コンピュータ・システムのオペレーティング・システムへのファイル・オペレーションを妨害し、

ここで、ファイル・オペレーションの妨害時に、前記ファイル変換モジュール・プログラムは、前記の少なくとも１つの変換データ・ファイルから変換データを検索し、前記の少なくとも１つの変換キーを使用して、前記変換データをその未変換状態に逆変換し、前記データを未変換状態で前記実行プログラムに送ることを特徴とする機械読み出しが可能な媒体。

【請求項２２】　請求項２１記載の媒体において、前記ファイル・オペレーションは、ファイル・オープン・ファイル・オペレーションと、ファイル読み出しファイル・オペレーションと、ファイル書き込みファイル・オペレーションと、ファイル・クローズ・ファイル・オペレーションとを含むことを特徴とする機械読み出しが可能な媒体。

【請求項２３】　請求項２２記載の媒体において、ファイル・オープン・ファイル・オペレーションを妨害するとき、前記ファイル変換モジュール・プログラムは、前記の少なくとも１つの変換データ・ファイルに対して独自のファイル識別子を割り当て、前記独自のファイル識別子を、アクティブ・ファイル・リスト中に記録することを特徴とする機械読み出しが可能な媒体。

【請求項２４】　請求項２３記載の媒体において、ファイル読み出しファイル・オペレーションまたはファイル書き込みファイル・オペレーションを妨害するとき、ファイル変換モジュール・プログラムは、アクティブ・ファイル・リストをチェックして、逆変換を行うべきかどうかを判断し、行わない場合には、前記ファイル・オペレーションを、通常処理用の前記オペレーティング・システムに戻すことを特徴とする機械読み出しが可能な媒体。

【請求項２５】　請求項２３記載の媒体において、コンピュータ・システムが走っているとき、ファイル変換モジュール・プログラムは、前記の少なくとも１つの変換データ・ファイルを閉じるときに、前記アクティブ・ファイル・リストから、前記の少なくとも１つの変換データ・ファイルに対する独自のファイル識別子を削除することを特徴とする機械読み出しが可能な媒体。

【請求項２６】　請求項２１記載の媒体において、前記の少なくとも１つの変換データ・ファイルは、読み出し専用ファイルであることを特徴とする機械読み出しが可能な媒体。

【請求項２７】　請求項２１記載の媒体において、前記の少なくとも１つの変換データ・ファイルは読み出し/書き込みファイルであることを特徴とする機械読み出しが可能な媒体。

【請求項２８】　請求項２１記載の媒体において、前記実行プログラムは、前記実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼と共に組み込まれ、前記プロテクト・モジュール・プログラムは、前記実行プログラムを

最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とする機械読み出しが可能な媒体。

【請求項２９】　請求項２８記載の媒体において、少なくとも１つの変換キーは、前記プロテクト・モジュール・プログラム内に組み込まれることを特徴とする機械読み出しが可能な媒体。

【請求項３０】　請求項２８記載の媒体において、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリ（ＤＬＬ）であることを特徴とする機械読み出しが可能な媒体。

【請求項３１】　請求項２１記載の媒体において、少なくとも１つの前記変換データ・ファイルは、ＤＥＳおよびＲＳＡ暗号化アルゴリズムのうちの１つを用いて暗号化されることを特徴とする機械読み出しが可能な媒体。

【請求項３２】　請求項２１記載の媒体において、非実行ライセンス・データを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とする機械読み出しが可能な媒体。

【請求項３３】　請求項２１記載の媒体において、実行プログラム・コードを前記実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記実行プログラムに確実に供給することを特徴とする機械読み出しが可能な媒体。

【請求項３４】　（ａ）少なくとも１つの変換データ・ファイルをストアするメモリと、（ｂ）オペレーティング・システム、修正実行プログラムを実行するプロセッサとを有し、

前記修正実行プログラムは、変換サービスを前記修正実行プログラムに供給するための変換モジュール・プログラムを起動するために修正され、前記ファイル変換モジュール・プログラムは、前記修正実行プログラムから前記オペレーティング・システムへのファイル・オペレーションを妨害し、

ここで、ファイル・オペレーションを妨害する際に、前記ファイル変換モジュール・プログラムは、少なくとも１つの変換データ・ファイルから変換データを検索し、変換キーを用いて、前記変換データを、その未変換状態に逆変換し、前記のデータを、未変換状態のまま、前記の修正実行プログラムに送ることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項３５】　請求項３４記載のシステムにおいて、前記ファイル・オペレーションは、ファイル・オープン・ファイル・オペレーションと、ファイル読み出しファイル・オペレーションと、ファイル書き込みファイル・オペレーションと、ファイル・クローズ・ファイル・オペレーションとを含むことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項３６】　請求項３５記載のシステムにおいて、ファイル・オープン・ファイル・オペレーションが行われる際、前記ファイル変換モジュール・プログラムは、少なくとも１つの前記変換データ・ファイルに対して、独自のファイル識別子を割り当て、前記独自のファイル識別子を、アクティブ・ファイル・リスト中に記録することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項３７】　請求項３６記載のシステムにおいて、ファイル読み出しおよびファイル書き込みファイル・オペレーションを行う際、前記ファイル変換モジュール・プログラムは、アクティブ・ファイル・リストをチェックして、逆変換を行うべきかどうかを決定し、行わない場合には、前記ファイル・オペレーションを、通常処理用の前記オペレーティング・システムに戻すことを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項３８】　請求項３６記載のシステムにおいて、前記変換データ・ファイルを閉じる際、前記ファイル変換モジュール・プログラムは、前記アクティブ・ファイル・リストから、少なくとも１つの変換データ・ファイルに対する前記独自のファイル識別子を検出することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項３９】　請求項３４記載のシステムにおいて、少なくとも１つの前記変換データ・ファイルは、読み出し専用ファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４０】　請求項３４記載のシステムにおいて、少なくとも１つの前記変換データ・ファイルは、読み出し/書き込みファイルであることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４１】　請求項３４記載のシステムにおいて、前記修正実行プログラムは、前記修正実行プログラムが実行されるときにだけ起動されるプロテクト・モジュール・プログラムを実行するための呼と共に組み込まれ、前記プロテクト・モジュール・プログラムは、前記修正実行プログラムを最初に実行するとき、前記ファイル変換モジュールを初期化することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４２】　請求項４１記載のシステムにおいて、前記変換キーは、前記プロテクト・モジュール・プログラム内にストアされることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４３】　請求項４１記載のシステムにおいて、前記プロテクト・モジュール・プログラムは、ダイナミック・リンク・ライブラリ（ＤＬＬ）であることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４４】　請求項３４記載のシステムにおいて、少なくとも１つの前記変換データ・ファイルは、ＤＥＳおよびＲＳＡ暗号化アルゴリズムのうちの１つを用いて

暗号化されることを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４５】　請求項３４記載のシステムにおいて、非実行ライセンス・データを前記修正実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記修正実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【請求項４６】　請求項３４記載のシステムにおいて、実行プログラム・コードを前記修正実行プログラム中に組み込み、前記ファイル変換モジュールが変換サービスを前記修正実行プログラムに確実に供給することを特徴とするソフトウェア・プログラム・プロテクト・システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、非認可のコンピュータ・ソフトウェア・プログラムの修正および/または再配布を防止するプロテクト機構に関するものである。

【０００２】

【従来の技術】商業分野のソフトウェア、特に、消費者向け個人コンピュータすなわちＰＣのソフトウェアの分野では、市場を拡大する目的で、ソフトウェアのデモンストレーション版を、無料で、または、廉価で配布することが一般的に行われている。ソフトウェア販売業者は、自分たちのソフトウェアを、多くの潜在的な購買者に見せるために、このようなソフトウェアのデモンストレーション版を多量に配布している。このようなことが可能なのは、低コストで配布しても、ソフトウェアを試用した結果、これを気に入って購買するユーザがいるために、十分に利益が上がるからである。このようなデモンストレーション版は、よくインターネットを介して配布されるが、一般的には、フロッピーディスクやＣＤ－ＲＯＭまたはＤＶＤ（多目的光ディスク）のような物理的な媒体の方がよく使用されている。このような物理的な媒体は廉価で、特定の対象者に簡単に届けることができる。また、このような物理的な媒体は、多くの消費者向けインターネット接続を介するダウンロード支援ソフトウェアとは異なり、販売用と同じく完全な機能を持つ大容量のデモンストレーション版を配布するために必要な十分な容量を有している。

【０００３】このようなデモンストレーション・ソフトウェア・プログラムは、以下の特性を持つ。すなわち、（ｉ）配布形式は、ソフトウェア・プログラムの完全機能版である、（ｉｉ）デモンストレーション・ソフトウェア・プログラムの配布されたコピーはすべて同一である、（ｉｉｉ）通常のインストールでは、このソフトウェア・プログラムは、デモンストレーション仕様を供給し、１つまたはそれ以上の制限（機能性、時間、ユーザの数など）がある、（ｉｖ）ユーザがこのソフトウェアの完全なライセンス版を取得できる手段が提供され、また、ユーザは、追加の媒体の配布を受けることなく、このデモンストレーション版を、完全版に変換できる手段が提供される、（ｖ）このデモンストレーション・ソフトウェア・プログラムは、特定のハードウエアに依存せず、通常の消費者向けＰＣ上で走る（また、完全機能版に切り替わる）。

【０００４】概して、デモンストレーション用に配布されるデモンストレーション版ソフトウェアの多くは、実際、完全版の機能をすべて含んでいる。そこに埋め込まれるソフトウェア・プロテクト機構は、完全版を購入するまでは、完全版へのアクセスが阻止されるように設計されている。これには２つの理由がある。まず、ソフトウェア出版社は、デモンストレーション用のソフトウェアと完全版のソフトウェアという異なるソフトウェアの開発を避けたいという希望があるからであり、次に、ソフトウェア出版社は、ユーザに、デモンストレーション版を完全版に変換する手段を提供することによって、ソフトウェアの完全版を、再度ユーザに配布する手間を省けるからである。

【０００５】

【発明が解決しようとする課題】これまでの説明から、このシステムは、市場拡大のための有効なツールであるが、一方、重大な不利益もある。このようなデモンストレーション媒体は、ソフトウェア海賊にとって、主要なソースとなってしまうからである。ソフトウェアの海賊は、通常、ソフトウェア出版社が、そのソフトウェアにどのようなプロテクト機構を埋め込んだとしても、その機構を回避するように、デモンストレーション版を修正することができる。ソフトウェアのデモンストレーション版がそのように修正されてしまうと、配布されたプログラムに隠れていたプロテクト機構は、出版社が気付かないうちに解除されてしまう。さらに、いったん、ソフトウェア海賊が、ソフトウェア出版社が採用したプロテクト機構の回避に成功すると、いわゆる「クラック」と呼ばれる海賊版パッケージが作られ、他の人が同様に出版社に代金を支払わずに、デモンストレーション版のソフトウェア・プログラムを完全版に変換するための手助けをすることになる。この「クラック」プログラムが広範囲に渡って無料配布されると、ソフトウェア出版社は、多大な損害を受けることになる。

【０００６】当分野において、デモンストレーション版のソフトウェア・プログラムを、海賊の攻撃から防衛するように設計されたプロテクト手段がいくつか知られている。このようなプロテクト技術は、典型的には、ソフトウェア・プログラムのバイナリ実行形式に対し、内部機能を付加することによって、デモンストレーション版に制限を加えている。このプロテクト技術によって、デモンストレーション版の制限を回避するように設計された非認可の修正版が検出され、その結果、例えば、特定

の画面が表示されるか、または、そのプログラムが自動的に実行されなくなることが好ましい。しかしながら、バイナリ実行プログラム（例えば、ウインドウズ™ のオペレーティング・システムで「.ＥＸＥ」の拡張子を有するファイル）は、通常、ソフトウェア・プログラム全体から見れば、その容量は非常に小さい。多くの消費者向けソフトウェア・アプリケーションにおいて、実行可能なファイルは、１メガバイトぐらいの大きさであるが、プログラムを実行するのに必要なデータ・ファイルおよびその他のファイルは、全体で、数百メガバイトを超える。このように、ソフトウェア・プログラムの中心的なバイナリ実行ファイルのみをプロテクトしても、実際には、デモンストレーション版ソフトウェア全体の非常に小さい部分しか海賊行為からプロテクトされないのである。

【０００７】この技術によってプロテクトされたソフトウェア・プログラムが無料配布されると、最も単純な形式の「クラック（解読）」プログラムの場合には、プロテクトされた中心的なバイナリ実行ファイルを、合法的に購入されたプロテクトされていない正式版の実行ファイルに置き換えることができる。この「クラック」形式は、高度な技術を必要としないため、簡単に生成できる。これは、ソフトウェア・プロテクト技術における重大な弱点となっている。

【０００８】他のソフトウェア・プロテクト機構として、暗号技術が使われている。暗号はコンピュータ・メッセージを一度だけ伝送するときには、十分に解決策となるが、ソフトウェア・プロテクト機構としては、概して、十分とは言えない。この技術においては、プロテクト・ソフトウェア・プログラムが走るたびにその鍵を用いなければならないので、暗号システムに必要な暗号キーは本質的に発見され易くなる。

【０００９】以上のことから、改良されたソフトウェア・プロテクト機構が必要となる。

【００１０】本発明の目的は、コンピュータ・ソフトウェア・プログラムに対する改良されたソフトウェア・プロテクト機構を提供することにある。本発明においては、ソフトウェア・プログラムの「アセット・ファイル」（すなわち、データ・ファイル）を変換し、コアー実行プログラム自身を修正するシステムが開示される。この方法では、修正された実行プログラムを再配布するときは、それに対応する変換されたアセット・ファイルも再配布される。非変換（すなわち、オリジナル形式）のアセット・ファイルは修正された実行ファイルと一緒でなければ（また、その逆も同様）、再配布は実現されない。

【００１１】

【課題を解決するための手段】プロテクトされたソフトウェア・アプリケーションを配布できる形態にするために、そのアプリケーションは、まず、主要な構造要素に分解される。すなわち、コアー実行プログラムとアセット・ファイル（読み出し専用データ・ファイルおよび/または読み出し/書き込みデータ・ファイル）に分解される。変換キーを用いて、変換機能はアセット・ファイルに適用され、変換データ・ファイルを作る。その後、変換キーは、後に使用するためにストアされる。その後、プロテクト・モジュール・プログラムの実行用の呼を、コアー実行プログラムに組み込むことによって、修正された実行プログラムが形成される。プロテクト・モジュール・プログラムは、前記の修正された実行プログラムを実行するときにのみ、要求され、他の状態のもとでは動作しない。その後、ファイル変換モジュール・プログラムが生成され、大量のソフトウェアコンポーネントに付加される。変換キーと変換データ・ファイルのリストは、プロテクト・モジュール・プログラムに任意に付加される。その後、大量のソフトウェアコンポーネント（変換されたデータ・ファイル、変換キー、修正された実行プログラム、プロテクト・モジュール・プログラムおよびファイル変換モジュール・プログラム）がユーザへの配布用媒体に変換される。

【００１２】ソフトウェア・プログラム・プロテクト・システム上のオペレーションにおいて、修正された実行プログラムがロードされる。このプログラムは、自動的に、このプロテクト・モジュール・プログラムを走らせる。このプロテクト・モジュールは、ファイル変換モジュールのオペレーションを起動し、修正された実行プログラムによってコンピュータ・オペレーション・システムになされた要求（例えば、ファイル読み出し、ファイル書き出し、ファイル・オープン、およびファイル・クローズなど）を妨害するように設計されている。ファイル変換モジュールは、変換キーを用いて、ファイル・オペレーション要求が変換データとして識別されたときだけ、要求された逆変換を実行する。

【００１３】本発明は、プロテクト・モジュール・プログラムを使用することによって、修正された実行プログラムを、変換データ・ファイルに接続する。このように、ファイル変換モジュールは、どのような場合にも、修正された実行プログラムが存在するときおよびそのプログラムの制御下にあるとき以外は、要求された逆変換を実行する。

【００１４】本発明の他の側面と特徴は、実施の形態および添付の図面を用いて説明することによって、当業者に明確になる。

【００１５】

【発明の実施の形態】図１は、プロテクト機構を持たない典型的なインストールおよびランタイム構造を示すブロック図である。このようなソフトウェア・プログラムは、通常、ＣＤ－ＲＯＭ１０または他の物理的な媒体で配布される。本発明においては、配布ソースはＣＤ－ＲＯＭ以外（例えば、ＤＶＤ，インターネットによるダウ

ンロードなど）でもよいが、便宜上、ＣＤ－ＲＯＭであると仮定する。

【００１６】インストール・プログラム２０は、ＣＤ－ＲＯＭ１０上で走り、パス１１を介してＣＤ－ＲＯＭ１０の内容を読み込み、それをユーザＰＣの内臓ディスクと、コアー実行プログラム３０と、多数のファイルの組４０，４１，５０および５１に対してコピーする（パス２１，２２，２３，２４および２５を介して）。コアー実行プログラム３０は、典型的には、コアー実行プログラム４１でサポートされる実行オペレーティング・システム・コードに依存する。このコアー実行プログラム４１は、更新されてもよいし、または、インストール・プログラム２０によって初めからインストールされてもよい。または、ユーザのＰＣにすでにインストールされていてもよい。一例として、マイクロソフト社のウインドウズ™ のＤｉｒｅｃｔＸ™ グラフィック機能がある。コアー実行プログラム３０も、典型的には、コアー実行サポート・ファイル４０に依存し、また、インストール・プログラム２０によってインストールされる。この例としては、ウインドウズ™ のオペレーティング・システム中で特定のアプリケーションを用いる１以上のダイナミック・リンク・ライブラリ（ＤＬＬ）がある。

【００１７】適切に運転するために、コアー実行プログラム３０は、典型的には、データ・ファイルと相互作用し、また、データ・ファイルを操作する。例えば、読み出し/書き込みファイル・セット５１は、ファイル・システム・インタフェース６１を介してアクセスされるように示されている。また、読み出し専用ファイル５０は、ファイル・システム・インタフェース６０を介してアクセスされるように示されている。ファイル・セット５０と５１は、「アセット・ファイル」として知られている。図１には、ソフトウェア・プロテクト・アプリケーションは示されていないので、ソフトウェア・アプリケーションは、容易にコピーされ、配布される。

【００１８】図２は、当分野で既知の典型的なソフトウェア・プロテクト機構を付加したソフトウェア・プログラムのインストールとランタイム構造を示すブロック図である。修正実行ファイル３０Ａとプロテクト・モジュール・プログラム３１（これ以降、「プロテクト・モジュール３１」と呼ぶ）を除いて、図２のすべてのコンポーネントは、図１に示すコンポーネントと同一である。コンポーネント３０Ａと３１は、これらがプロテクトされていることを視覚的に分るように、破線で囲まれている。

【００１９】図２の修正実行ファイル３０Ａは、ファイルの完全性をプロテクトするために、図１のオリジナルのコアー実行プログラム３０にいくらか修正を加えたものである。一般に、付加的なプロテクト・サポート・モジュール実行コンポーネント（「プロテクト・モジュール」）３１は、パス２６経由でインストールされ、修正実行プログラム３０Ａの制御下において、ランタイム６２で使用される。プロテクト・モジュール３１は、必ずしも、プログラム実行中に使用可能ではないが、典型的には、プログラムがデモンストレーション・モードにある間、修正実行プログラム３０Ａを監視し、および/または修正実行プログラム３０Ａの機能をセットアップすることによって、予め設定された制限に合うようにする。

【００２０】図２で示すシステムの弱点は、もし、プロテクト実行プログラム３０Ａがオリジナルのコアー実行プログラム３０のコピーによって置き換えられると、すべてのプロテクトが失われ、その結果、このアプリケーションは、非認可の完全機能版と同じになる。そのコンポーネントがまだ存在していたとしても、プロテクト・モジュール３１によって提供されるプロテクト機構は失われてしまう。というのは、このようなプロテクト機構は、修正実行プログラム３０Ａによってのみ呼び出され、もし、プロテクトされていないコアー実行プログラム３０が修正実行プログラム３０Ａの代わりに使用されれば、起動されることはなく、休止状態のままである。

【００２１】図３は、アセット・ファイル用ソフトウェア・プロテクト機構が追加された、ソフトウェア・プログラムのインストールおよびランタイム構造を示すブロック図である。図３において、コンポーネント２０，４０，４１のみが、図１および図２で示すコンポーネントと同じである。ＣＤ－ＲＯＭ１０Ａを除いて、残りのコンポーネントは破線の輪郭線で示される。これは、これらのコンポーネントが、本発明のソフトウェア・プロテクト機構に従ってプロテクトされていることを意味している。ＣＤ－ＲＯＭ１０Ａは、プロテクトされたソフトウェア用の配布媒体である。図２で示すように、修正実行プログラム３０Ａは、ファイルの完全性をプロテクトするために、図１のオリジナルのコアー実行プログラム３０にいくらか修正を加えたものである。付加的なプロテクトサポート・モジュール実行コンポーネント（「プロテクト・モジュール」）３１は、パス２６経由でインストールされ、修正実行プログラム３０Ａの制御下で、ランタイム６２で使用されるのは、図２と同じである。

【００２２】図３において、図２のファイル・セット５０と５１は、変換ファイル・セット５０Ａと５１Ａによって置き換えられている。変換ファイル・セット５０Ａと５１Ａ中には、１以上の変換データ・ファイルがあってもよい。変換ファイル・セット５０Ａと５１Ａは、実行ファイルを含む任意のファイルのフォーマットを含んでもよい。変換ファイル・セット５０Ａと５１Ａは、インタフェース読み出し/書き込み機能６１Ａとインタフェース読み出し専用機能６０Ａを介して（また、図４に示される変換モジュール７０を介して）、修正実行プログラム３０Ａによってアクセスされる。これらのインタフェース読み出し/書き込み機能６０Ａとインタフェー

ス読み出し専用機能６１Ａは、適切なランタイム変換と逆変換を提供する。変換と逆変換の正確な特徴は本発明の基本的な部分でないことは、当業者であれば理解できる。例えば、変換には、暗号化や、データ暗号化標準（ＤＥＳ）またはリベスト・シャミール・アルドレマン（ＲＳＡ）のような標準アルゴリズムを用いた復号化、または他のもっと単純な暗号化および復号化アルゴリズムが含まれる。また、同じアプリケーション中で異なる変換の混合も使用される。唯一、実際に要求されることは、変換に可逆性があることであり、その変換が、ソフトウェア・プログラムの実行と変換の間で適切なバランスを提供することである。例えば、単純な変換では、秘密バイト値を用いて、ビット・シフトまたは排他的オア・オペレーションのようなバイトごとの論理オペレーションを変換バッファ・データに適用し、クリア・テキスト・データを作ってもよい。インタフェース機能６０Ａと６１Ａは、要求されたランタイム変換を提供し、種々の方法で実行される。例えば、要求された変換は、ソース・コードで組み込まれてもよく、または修正実行プログラム３０Ａに対するライブラリ・レベルで組み込まれてもよい。これらも、オペレーティング環境によって固有に提供されてもよく、または特別な媒体と関連してもよい。

【００２３】図４は、本発明の一実施の形態のソフトウェア・プロテクト機構を示すブロック図である。図４は、変換ファイル・セット５０Ａと５１Ａ、修正実行プログラム３０Ａ、およびプロテクト・モジュール３１を示している。これらはすべて、図３中の対応するコンポーネントと同様である。図３のコンポーネントのうちいくつかは、見やすくするために、図４では省略されているが、論理的には、図４のシナリオにおいても、同じように存在している。インタフェース読み出し専用機能６０Ｂ，６０Ｃおよび６０Ｄ並びにインタフェース読み出し/書き込み機能６１Ｂ，６１Ｃおよび６１Ｄは、本発明による適切なランタイム変換と逆変換を提供する。

【００２４】本実施の形態において、インタフェース機能６０Ｂ、６０Ｃ、６０Ｄ、６１Ｂ、６１Ｃおよび６１Ｄは、修正実行プログラム３０Ａに対し透明である。すなわち、変換ファイル・セット５０Ａと５１Ａを用いる修正実行プログラム３０Ａ内のコードは、そのオリジナル・フォームのままである。なぜなら、本発明によれば、そのコードによって見られるファイルの入力／出力は、プロテクトされていない場合に見られるファイル入力／出力と同一であるからである。これは、オペレーティング・システム７１とファイル変換モジュール・プログラム７０（以下、「ファイル変換モジュール７０」と呼ぶ）によってなされる。オペレーティング・システム７１とファイル変換モジュール７０は、インタフェース６０Ｂ、６０Ｃ、６１Ｂ、６１Ｃを介して、ファイル・オペレーションの読み出しおよび書き込みを仲介する。オペレーティング・システム７１（例えば、マイクロソフト・ウインドウズ９５™またはマイクロソフト・ウインドウズ９８™）およびファイル変換モジュール７０は、そのファイル変換がファイル・オペレーションに透明に付加される手段である。

【００２５】ファイル変換モジュール７０を実行するには、種々の方法がある。しかしながら、本発明による本実施の形態によれば、このモジュールの目的は、（ｉ）修正実行プログラム３０Ａに対して、ファイル変換を完全に透明にすること、すなわち、ファイル変換の結果、このプログラムには何の修正も加えられないこと、（ｉｉ）適切な環境下でのみ、この場合は、合法的な修正実行プログラム３０Ａに対してのみに、ファイル変換サービスを提供することの２点である。変換は、オリジナル・コアー実行プログラム３０のような他の実行プログラム、または関連のないプログラムに所属するファイルのような影響を受けないアセットに対して、実行されることはない。これらの目的を満たす手段は、以下に述べるように、ファイル変換モジュール７０の論理によって初期化および実行される。

【００２６】ファイル変換モジュール７０のオペレーション論理は、図５、図６および図７のフローチャートに示される。オペレーションには明確な３つの位相がある。すなわち、初期化処理（図５）、ファイル・フック処理（図６）および終了処理（図７）である。

【００２７】ファイル変換モジュール７０は、オペレーティング・システム７１からのファイル・アクティビティが妨害される前に初期化されなければならない。プロテクト・モジュール３１は、修正実行プログラム３０Ａがあるときにだけ、アクティブである。例えば、ウインドウズ™・オペレーティング・システムに対しては、プロテクト・モジュール３１は、ダイナミック・リンク・ライブラリ（ＤＬＬ）ファイルであってもよい。このＤＬＬファイルは、修正実行プログラム３０Ａが生成される変換プロセスの一部として、オリジナルの実行プログラム３０の「インポート・テーブル」に付加されてもよい。これは、修正実行プログラム３０Ａの実際のランタイム・バイナリ命令が実行される前に、修正実行プログラム３０Ａが最初に起動されたとき、自動的に、ウインドウズ™・オペレーティング・システム中の「ローダ」の固有の動作を介して、プロテクト・モジュール３１中のランニング特定コードの効果を持つようになる。他のプログラムで、プロテクト・モジュール３１を走らせるものはない。プロテクト・モジュール３１は、ここで論じられるように、呼を介して、変換モジュール７０をセットアップするために、ファイル変換モジュール７０は、他のすべてのプログラムに対してサービスを提供するためにセットアップされることはない。オリジナルのコアー実行プログラム３０は、プロテクト・モジュール３１を起動しないばかりでなくそれがトリガする変換サ

ービスを起動しない。

【００２８】図５のフローチャートで示すとおり、ステップ５０１と５０２で、プロテクト・モジュール３１は、まずファイル変換モジュール７０を初期化する。この段階で、情報は、プロテクト・モジュール３１を介して、ファイル変換モジュール７０に送られる。このような情報には、アセット・リスト（すなわち、変換ファイル・セット５０Ａと５１Ａのメンバー）、呼プロセス識別情報、変換キー、および変換アルゴリズムのような他の選択的情報が含まれる。呼プロセス情報は、オペレーティング・システム７１から得られ、アセット関連情報は、プロテクト・モジュール３１に埋め込まれるか、または、他のファイルを読み出すことによって、プロテクト・モジュール３１から得てもよい。アセット関連情報も、他の手段によって、ファイル変換モジュール７０から得てもよい。例えば、これは、ファイル変換モジュール７０によって直接ファイルから読み出されるか、または、ファイル変換モジュール７０自身に埋め込むこともできる。本実施の形態において、ステップ５０３で、プロテクト・モジュール３１は、利用可能なオペレーティング・システム・サービスを用いて、ファイル変換モジュール７０を、ファイル・システムに「フック（妨害）」する。例えば、ウインドウズ９５™またはウインドウズ９８™では、ファイル変換モジュール７０は仮想装置ドライバ（ＶｘＤ）であってもよく、プロテクト・モジュール３１は、ウインドウズ™の「仮想ファイル・システム」呼を用いて、そのオペレーティング・システムが、ファイル変換モジュール７０を後続のファイル・システム呼上で実行されるオペレーションの低レベル・シーケンスの一部として起動するように調整してもよい。他の例として、１３・ヘキサデシマル中断用の「終端および常駐」（ＴＳＲ）プログラム中のソフトウェア中断ハンドラは、マイクロソフト・ディスク・オペレーティング・システム（ＭＳ－ＤＯＳ™）に対して使用することができる。図４はこのフック（妨害）機能を示し、この機能では、オペレーティング・システム７１に対するインタフェース読み出し／書き込み機能６１Ｂとインタフェース読み出し専用機能６０Ｂは、インタフェース機能６０Ｃと６１Ｃで、ファイル変換モジュール７０によって妨害される。本実施の形態において、このような妨害は、ファイル読み出し、ファイル書き込み、ファイル・オープン、またはファイル・クローズ・オペレーションが実行されるときはいつでも、行われる。初期化プロセスは、その後、ステップ５５３で完了する。フックが確立され、ファイル変換モジュール７０が初期化されると、ファイル変換モジュール７０は、後続のファイル要求によって自動的に起動される。

【００２９】ファイル変換モジュール７０は、いったん初期化されると、オペレーティング・システム７１に送られたファイル・オペレーティング要求を受け取る。オペレーティング・システム７１への要求が、修正実行プログラム３０Ａへの、または修正実行プログラム３０Ａからの変換データに対してのものであると識別されるときにのみ、ファイル変換モジュール７０は、要求された逆変換（すなわち、「変換サービス」）を実行する。もし、ファイル変換モジュール７０が、その要求が修正実行プログラム３０Ａに向けられたものでもなく修正実行プログラム３０Ａからのものでもないと判断すれば、妨害要求は、通常の処理をするために、オペレーティング・システム７１に戻される。あらゆる場合において、ファイル変換モジュール７０は、オペレーションに関するファイル・データ（もし、あれば）が、ファイル変換モジュール７０に知られているメモリ・バッファ中でオペレーティング・システム７１によってすでに置き換えられるように起動される。その後、ファイル変換モジュール７０は、このバッファ・データに変換を行うオプションを有するか、または、単に、オペレーティング・システムに戻り、データを発見されたときの状態のままにしておく。

【００３０】ファイル変換モジュール７０に関するファイル・オペレーションのセットは、本発明の異なる実行に従って変化する。少なくとも、ファイル・オープン、ファイル読み出し、およびファイル・クローズ・オペレーションとは関連がある。ファイル書き込みオペレーションを含む他のオペレーションも、ファイル変換モジュール７０を起動する同じオペレーティング・システム・フックを介して、トリガとして使用される。

【００３１】これらのファイル・オペレーションに関するさらなる情報を以下に説明する。

（ｉ）　ファイル・オープン：　マイクロソフト・ＤＯＳ™、マイクロソフト・ウインドウズ™およびユニックス™のようなオペレーティング・システム環境は、整数ファイル識別子の概念を使用して、オープン・ファイルを追跡する。プロセス・アセット・リスト（すなわち、変換ファイル・セット５０Ａと５１Ａ）にリストされているファイルに対し、安全プロセス（すなわち、修正実行プログラム３０Ａ）から発生したファイル・オープン要求によって、ファイル変換モジュール７０は、関連変換キーによって記録されるべきオープン・ファイル用にオペレーティング・システム７１によって指定された独自のファイル識別子を記録し、このファイルを、アクティブ・ファイル・リストに付加する。

（ｉｉ）　ファイル読み出し：　上記のファイル・オープン中に確立された独自のファイル識別子を有するファイル読み出しオペレーションが、プロテクトされたアセット・ファイルの独自の識別子に対して起こるとき、オペレーティング・システム７１から戻されたデータは、ファイル変換モジュール７０によって復号化される。これは、オペレーティング・システム７１に、呼プロセスによって特定された宛先バッファへの読み出しオペレー

ションを完了させることによって、また、引き続き、アセット・ファイルの関連キーによってデータを適切に（すなわち、同じ宛先バッファ中で）変換させることによって達成される。

（ｉｉｉ）　ファイル書き込み：　図６には、ファイル書き込みオペレーションの処理は示されていないが、これはファイル・オープン・オペレーションに類似している。変換モジュール７０は、オペレーティング・システム環境を呼び出す前に、呼プロセスのデータ・バッファ中で、データを適切に逆変換して、実際のファイル書き込みオペレーションを実行する。

（ｉｖ）　ファイル・クローズ：　変換モジュール７０がそのファイルを閉じる要求を受け取ると、オープン・プロテクト・アセット・ファイル用のファイル識別子とキーをストアするために使用されるリソースは、破棄される。

【００３２】図６は、本発明のファイル変換モジュール７０で行われるオペレーションのステップを表わすフローチャートである。ステップ５１０で、ファイル変換モジュール７０への「フック」は、オペレーティング・システム７１によって起動される。ファイル変換モジュール７０は、その要求が修正実行プログラム３０Ａへ（または、から）のものでないと判断すると、妨害要求は、通常の処理をするために、ステップ５２４と５２５で、オペレーティング・システム７１に戻される。

【００３３】しかしながら、呼プロセスがプロテクトされている（つまり、要求が、修正実行プログラム３０Ａへ（または、から）の変換データに対するものである）場合には、ファイル・オペレーションの解析は、ステップ５１２で行われる。図６において、実行すべきオペレーションは、ファイル・オープン、ファイル・クローズ、およびファイル読み出しのオペレーションである。「ファイル書き込み」オペレーションを含む他のオペレーションも、同様に、妨害することができる。

【００３４】オペレーション・モードがファイル読み出しのとき、ステップ５１６で、ファイル識別子が、アクティブ・ファイル・リスト中にあるかどうかが判断される。もしなければ、妨害された要求は、ステップ５１７とステップ５２５で、通常処理のため、オペレーティング・システム７１に戻る。ファイル識別子が、アクティブ・ファイル・リスト中にあれば、ファイル変換モジュール７０は、ステップ５２１で、オペレーティング・システム７１に戻り、オペレーティング・システム７１は、通常の方法で低レベル読み出しを完了する。これは、オペレーティング・システム７１が、ステップ５２２で、ファイル変換モジュール７０の制御に戻る（図示されてはいないが、ファイル変換モジュール７０の論理を介して）ような方法で行われる。このアセットに関する正しい変換キーは、ステップ５２２で決定され、読み出しバッファ中のデータは、ステップ５２３で変換される。前記のデータは、「フック」された読み出しオペレーションの通常の方法で、オペレーティング・システム７１によって、そこに置かれ、ファイル変換モジュール７０を起動する。この処理は、その後、ステップ５２５で完了する。

【００３５】オペレーション・モードがファイル・クローズのとき、ステップ５１８で、ファイル識別子がアクティブ・ファイル・リスト中にあるかどうかが判断される。もしなければ、妨害された要求は、ステップ５１７とステップ５２５で、通常処理のためにオペレーティング・システム７１に戻る。ファイル識別子がアクティブ・ファイル・リスト中にあれば、ステップ５１９で、ファイル識別子と変換キーは、アクティブ・ファイル・リストから除去される。ステップ５２０で、ファイル・クローズ・オペレーションが実行される。この処理は、その後、ステップ５２５で完了する。

【００３６】図７は、本発明の終了処理ステップのフローチャートである。この終了ステップにおいては、ステップ５５１で、呼プロセスに割り当てられたすべてのリソースが解放され、ステップ５５２で、オペレーティング・システム環境フック機構が除去される。その後、終了処理ステップはステップ５５３で完了する。

【００３７】図８は、本発明の方法に従って、ソフトウェア・プログラムがアセットを変換するプロセスを示すブロック図である。これは、ソフトウェア・プログラムを配布状態に変換する前段階の配布変換プロセスである（図４参照）。まず、ソフトウェア・プログラムのオリジナルの配布形態１０は、プロセス１００によって、キーコンポーネント（すなわち、インストール・プログラム２０，読み出し専用ファイル・セット５０，読み出し/書き込みファイル・セット５１，コアー実行サポート・ファイル４０，一般実行システム・リソース４１，およびオリジナル・コアー実行プログラム３０）に分解される。

【００３８】その後、ソフトウェア変換プログラム１０１が実行される、この論理は、図９のフローチャートに示される。このソフトウェア・プログラムは、典型的には、ユーザ入力によって、ランタイム非実行コンポーネントである読み出し専用ファイル・セット５０と読み出し/書き込みファイル・セット５１から、特定のアセット・プログラムを選択する。ソフトウェア変換プログラム１０１は、選択されたカアセット・ファイルごとに、変換アルゴリズムとキーを選択する。その後、ファイルごとに選択された暗号化アルゴリズムは、その特定のファイル用のキーを使用して適用される。

【００３９】オリジナル・コアー実行プログラム３０も、修正が必要である。特に、オリジナル・コアー実行プログラム３０は、プロテクト・モジュール３１を起動する修正実行プログラム３０Ａに変換される。例えば、ウインドウズ™ 環境では、これは、オリジナル・コアー

実行プログラム３０の入力テーブルの拡張版中のＤＬＬに、参照を付加することによって達成される。プロテクト・モジュール３１およびファイル変換モジュール７０は、ファイル・セットに付加されなければならない。これらは、個別のファイルとしてストアされるか、または、プロテクトの理由から、ファイル３０Ａ，３１および/または７０のような他のファイルに組み込まれる。

【００４０】最終的には、非ランタイム・コンポーネントにも、修正が必要となる。特に、ソフトウェア・プログラムのインストール・プログラム２０は、上述の新しいコンポーネントおよび変化したコンポーネントを含むように強化される必要があり、その結果、修正インストール・プログラム２０Ａとなる。

【００４１】このプロセスが終了すると、ランタイム・コンポーネントの新しいセット（すなわち、修正実行プログラム３０Ａ，プロテクト・モジュール３１，ファイル変換モジュール７０，および変換ファイル・セット５０Ａと５１Ａ）が存在するようになる。これらのコンポーネントは、その後、ソフトウェア・ユーザに配布するために、プロセス１０２を用いて、ＣＤ－ＲＯＭまたは他の媒体に変換される。これらのコンポーネントは、それ以後、アプリケーション・ランタイムで必要に応じて起動される。

【００４２】本発明にとって、前述の修正が、すべて、ソフトウェア変換プログラム１０１によって実行されるかどうか、または、オリジナル・コアー実行プログラム３０および/またはインストール・プログラム２０上で実行されるような変換が、独立ではあるが関連したプロセスによって実行されるかどうかは重要ではない。また、厳密には、修正実行プログラム３０Ａがファイル変換モジュール７０を（通常は、プロテクト・モジュール３１を介して）起動するかぎり、他の変換のどのような性質を用いて修正実行プログラム３０Ａを生成するかも重要ではない。

【００４３】コアー実行プログラム３０を、修正実行プログラム３０Ａに変換するには、他の方法もある。これらの方法も、ファイル変換モジュール７０の変換を有効に制御できる。例えば、修正実行プログラム３０Ａを生成する変換プロセスは、非実行データを、コアー実行プログラム３０に付加する。この結果、このデータは、ファイル変換モジュール７０によって、ランタイムで、発見され検査される。このデータは、「ライセンス」の形式として、また、この特定のプログラムに変換を加えるかどうかを決定する際に、ファイル変換モジュール７０によって使用される存在およびコンテンツとして機能する。別の変換機能として、この変換プロセスは、コアー実行プログラム３０に、他の実行「コールバック」機能を付加することができる。この機能は、前に存在するプログラム３０の実行コードには何の関係も必要としないが、前記のモジュールが、変換サービスを供給するかどうかを判断するとき、ファイル変換モジュール７０によって呼ばれる。前述した種々の実施の形態においては、ファイル変換モジュール７０の最初の初期化は、コアー実行プログラム３０によって（または、もしあれば、プロテクト・モジュール３１によって）起動されることはないが、アプリケーション・インストール・プログラム２０のようないくつかの代替手段によって初期化される。

【００４４】図９は、図８で示すファイル変換プロセスのフローチャートである。ステップ６０１で、ファイル変換プロセスは開始される。ステップ６０２で、ユーザは、新しいランタイムおよび非ランタイム・コンポーネントの新たなセット（すなわち、修正実行プログラム３０Ａ，プロテクト・モジュール３１，ファイル変換モジュール７０，変換ファイル・セット５０Ａと５１Ａ，および修正インストール・プログラム２０Ａ）に対する宛先ディレクトリを選択する。ステップ６０３で、ユーザは、本発明に係る変換されるべきコアー実行ファイル（実行可能な）を選択する。ステップ６０４で、ユーザは、また、変換されるべきデータ・ファイル（「アセット・ファイル」としても知られている）を選択する。ステップ６０５で、ソフトウェア変換プログラムは、変換キーを生成する。ステップ６０６で、選択されたアセット・ファイルは、変換キーを使用して変換され、宛先ディレクトリ中に置かれる。決定ステップ６０７を使用して、ステップ６０４から６０６は、これ以上、変換するべきアセットがなくなるまで、繰り返される。この点において、アセット・リストと関連変換キー・セットが使用され、他のファイルに別々にストアされ、または組み込まれる。

【００４５】ステップ６０８で、プロテクト・モジュール３１が検索され、前述のアセット・ファイルと変換キー情報が選択的に付加される。ステップ６０９で、修正実行プログラム３０Ａは、プロテクト・モジュール３１に依存して生成される。前述のように、そのような依存は、ウインドウズ™の環境では、典型的には、プロテクトＤＬＬを参照する修正実行プログラム３０Ａに、インポート・テーブル参照を付加することである。この発明の一実施の形態においては、オリジナル・コアー実行プログラム３０をプロテクト・モジュール３１にリンクすることは、オリジナル・コアー実行プログラム３０になされる唯一の修正であり、それによって修正実行プログラム３０Ａを生成する。ファイル変換プロセスは、ステップ６１０で完了する。

【００４６】以下は、本発明の一実施の形態によるオペレーションの一例である。

【００４７】１．ソフトウェアの出版元は、ウインドウズ９５環境用のソフトウェア・プログラムを作っている。この一例として、種々のレベルを備えたロールプレイ・エンターテインメント・プログラムがある。この例

では、各レベルは、５０メガバイトの読み出し専用データ・ファイル（アセット）で示される。コアー実行ファイルは、２メガバイトの.ＥＸＥファイルである。

【００４８】２．本発明では、図８に示すように、各データ・ファイル５０が変換データ・ファイルに変換され、これらの変換データ・ファイルに依存して、修正実行プログラム３０Ａが生成されるように、ソフトウェア・プログラムが変換される。

【００４９】３．このソフトウェア・プログラムの変換形式は、パッケージされて、ＣＤ－ＲＯＭまたは他の媒体によってユーザに配布される。

【００５０】４．ユーザは、ソフトウェア・プログラムをインストールし、走らせる。

【００５１】５．インストールの際、修正実行プログラム３０Ａ、修正読み出し専用ファイル・セット５０Ａ、プロテクト・モジュール３１およびファイル変換モジュール７０は、すべてコピーされるか、または、ユーザのコンピュータで使用される。これは、ソフトウェア・プログラムに対して通常使用されるすべてのコンポーネントとは別なものである。プロテクト・モジュール３１は、オブジェクト・リンクおよび組み込み（ＯＬＥ）オブジェクト、個別の実行ファイル、またはダイナミック・リンク・ライブラリ（ＤＬＬ）ファイルの形式を取る。ウインドウズ９５™、またはウインドウズ９８™において実行可能なファイルの場合、ファイル変換モジュール７０としては、ウインドウズ™仮想デバイス・ドライバ（ＶｘＤ）が最適である。本実施例においては、ソフトウェア・プログラムの読み出し/書き込みファイルは影響されることはない。

【００５２】６．プログラムが走ると、まず、ユーザに第１レベルのソフトウェア・プログラムが供給される。これは、第１のレベルのデータ・ファイルが、必ず、修正実行プログラム３０Ａによって、アクセスされ読まれなければならないことを意味する。図６のフローチャートによれば、最初のファイル・オープン・オペレーションは、ファイル変換モジュール７０をトリガする。このファイル変換モジュール７０は、ファイルとファイル読み出しプログラムの特定の組み合わせで変換が行われることを決定する。

【００５３】７．レベル１のデータ・ファイルに対するそれ以降のすべてのファイル読み出しは、ファイル変換モジュール７０を起動する。図６によると、ファイル変換モジュール７０は、ファイル・システムに、実際の読み出しを行わせ、バッファ・ファイル・データを、使用される前に適切に変換し、オペレーションシステム７１によって、修正実行プログラム３０Ａに変換する。このことは読み出し/書き込みファイルに対しては一般的ではないが、書き込みは、同様に妨害され、必要な場合には変換される。

【００５４】８．プログラムが終了するか、または、ユーザがソフトウェア・プログラム内の他のレベルに移動すると、レベル１のファイルは閉じる。そのとき、変換モジュール７０は、そのデータ・ファイルに関連するすべての情報をリセットする。本発明によれば、そのファイルのそれ以降のすべての操作は、ローカル・ファイル・システムと変換モジュール７０の両方によって要求された時に、ファイル・オープン・オペレーションによって開始される。

【００５５】今まで、本発明の好適な実施の形態について説明してきたが、本発明の精神から逸脱することなく変形および改良ができるので、本発明は限定的に解釈されるべきではない。本発明の範囲は、以下の請求の範囲とそれに関連する明細書の記載によって定義される。

【図面の簡単な説明】

【図１】　プロテクト機構を持たないソフトウェア・プログラムの典型的なインストールとランタイムを示すブロック図である。

【図２】　典型的なソフトウェア・プロテクト機構を付加したソフトウェア・プログラムのインストールとランタイムを示すブロック図である。

【図３】　アセット・ファイル用のソフトウェア・プロテクト機構を付加したソフトウェア・プログラムのインストールとランタイムを示すブロック図である。

【図４】　本発明の一実施の形態によるソフトウェア・プロテクト機構を示すブロック図である。

【図５】　本発明の初期化ステップのフローチャートである。

【図６】　本発明の変換モジュールによるオペレーション・ステップのフローチャートである。

【図７】　本発明の終了ステップのフローチャートである。

【図８】　本発明の方法により、ソフトウェア・プログラムがアセットを変換するプロセスを示すブロック図である。

【図９】　図８で示されるファイル変換プロセスのフローチャートである。

【符号の説明】

１０、１０Ａ…ＣＤ－ＲＯＭ、
１１…パス
２０…インストール・プログラム、
２０Ａ…修正インストール・プログラム、
２１～２６…パス
３０…コアー実行ファイル（または、オリジナル・コアー実行プログラム）、
３０Ａ…修正実行ファイル（または、修正コアー実行プログラム、プロテクトプログラム）、
３１…プロテクト・モジュール・プログラム（または、プロテクト・モジュール）、
４０…コアー実行サポート・ファイル、
４１…実行オペレーティング・システム・コード（また

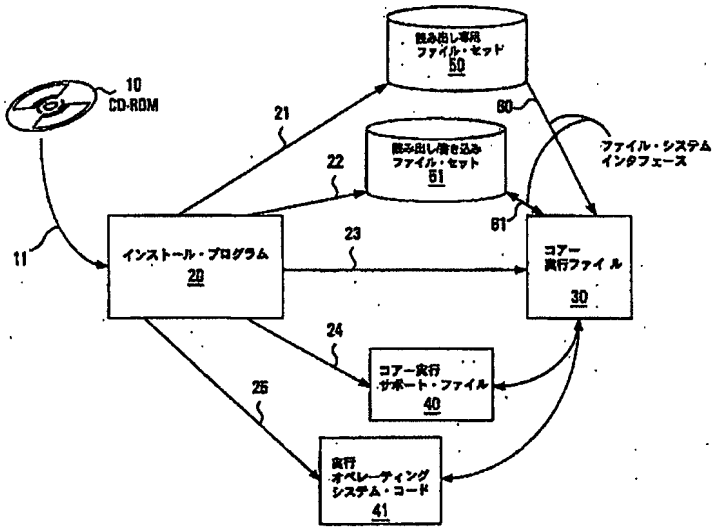
23

は、コアー実行プログラム、一般実行システム・リソース）、
５０…読み出し専用ファイル・セット（または、データ・ファイル）、
５０Ａ…変換ファイル・セット（または、修正読み出し専用ファイル・セット）、
５１…読み出し/書き込みファイル・セット、
５１Ａ…変換ファイル・セット、
６０，６１…ファイル・システム・インタフェース、＊

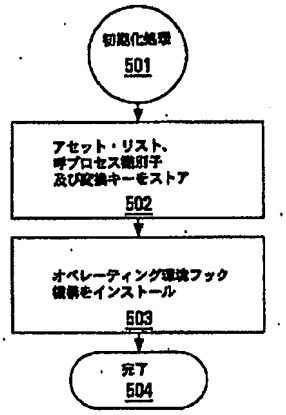
24

＊６０Ａ，６０Ｂ，６０Ｃおよび６０Ｄ…インタフェース読み出し専用機能、
６１Ａ，６１Ｂ，６１Ｃおよび６１Ｄ…インタフェース読み出し/書き込み機能、
７０…ファイル変換モジュール・プログラム、
７１…オペレーティング・システム、
１００…プロセス
１０１…ソフトウェア変換プログラム

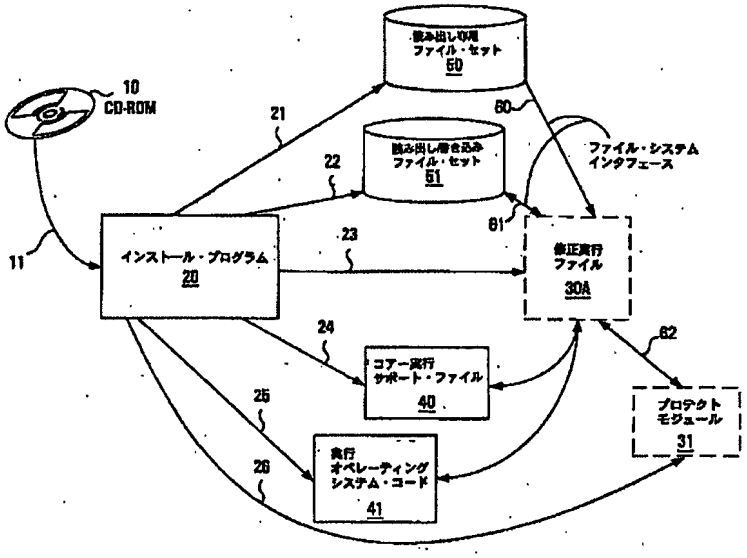
【図１】



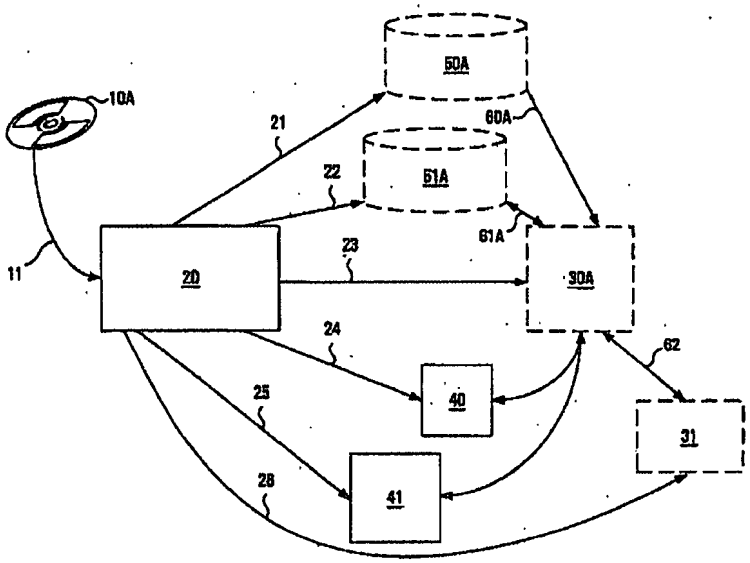
【図５】



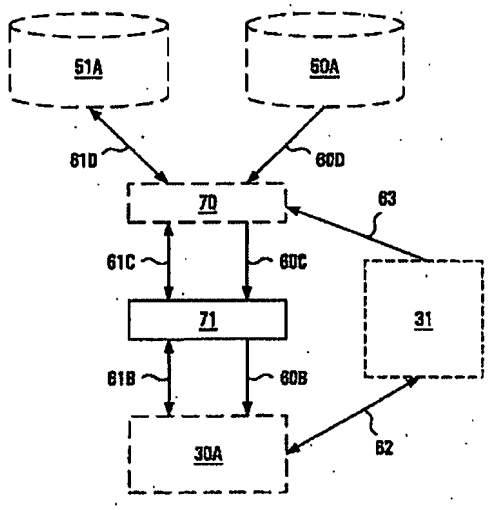
【図２】



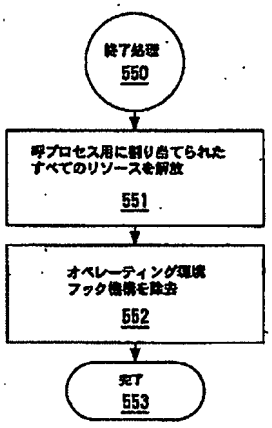
【図3】


50A
10A
21
60A
22
51A
61A
11
20
23
30A
24
62
40
25
31
41
28

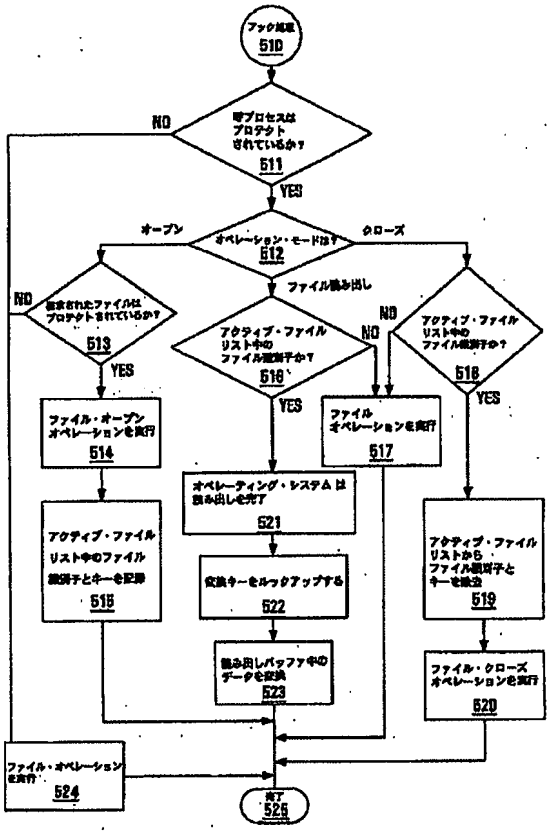
【図4】


51A
50A
61D
60D
70
63
61C
60C
71
31
61B
60B
30A
62

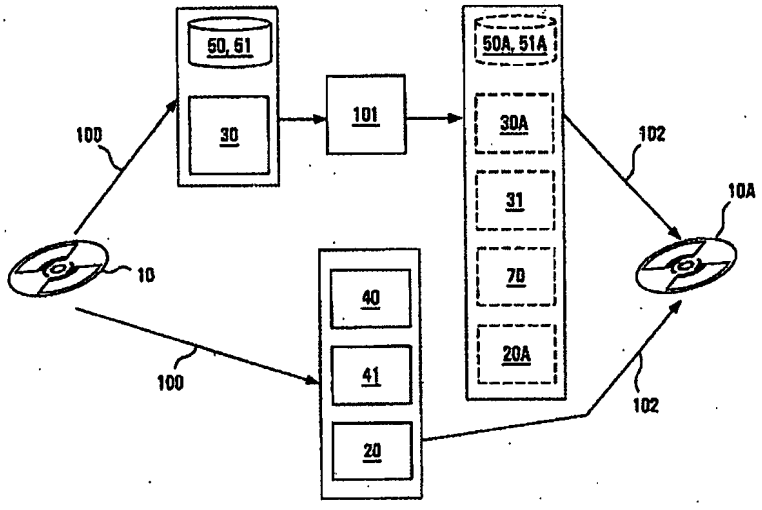
【図7】


終了処理
550
呼プロセス用に割り当てられた
すべてのリソースを解放
551
オペレーティング環境
フック機構を除去
552
完了
553

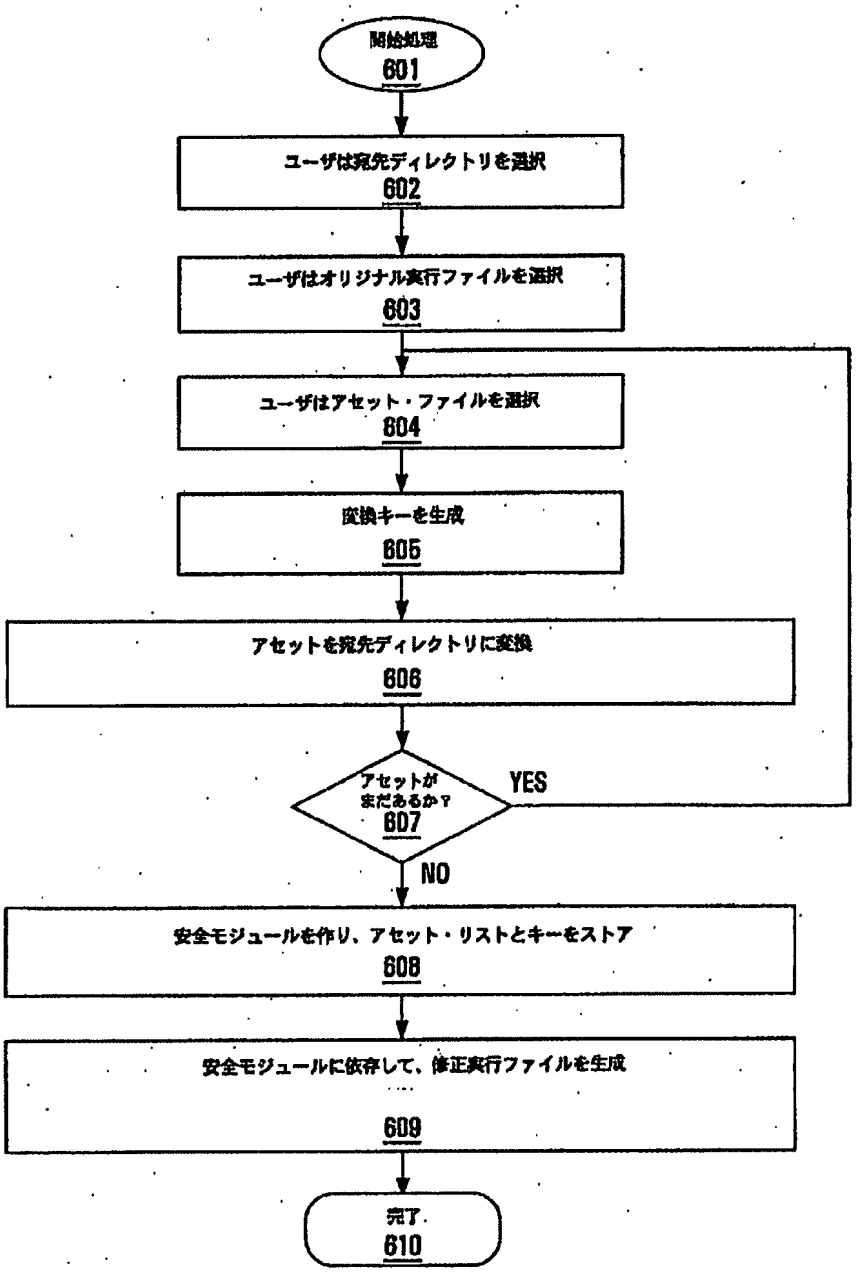
【図６】


510
511
NO
YES
512
オープン
クローズ
ファイル読み出し
513
NO
YES
514
515
516
NO
YES
517
518
NO
YES
519
520
521
522
523
524
525

【図８】


50, 51
30
101
50A, 51A
30A
31
70
20A
40
41
20
100
100
102
102
10
10A

【図９】



フロントページの続き

(72)発明者 スコット・アラン・トムソン
カナダ国，ケイ２エイ　１ティー９，オンタリオ，オタワ，エディソン　アベニュー
478